Stat Comp HW#4

Jason Rights

October 24, 2015

Problem 1:

```
secant <- function(fun, x0, x1, tol=1e-7, iter=100){
    for(i in 1:iter) {
        x2 <- x1-fun(x1)*(x1-x0)/(fun(x1)-fun(x0))
        if (abs(fun(x2)) < tol)
            return(x2)
        x0 <- x1
        x1 <- x2
    }
    stop('method did not converge')
}

f <- function(x) cos(x) - x

start.time.sec <- proc.time()
secant(f, x0=1, x1=2)
```

```
## [1] 0.7390851
```

```
total.time.sec <- proc.time() - start.time.sec
print(total.time.sec)
```

```
##    user  system elapsed
##       0       0       0
```

```
newton <- function(x, f, fp, tol=1e-7, iter=100){
  i <- 1
  while(abs(eval(f)) > tol & i < iter) {
    x <- x - eval(f)/eval(D(f, 'x'))
    i <- i + 1
  }
  if(i == iter) {
    stop('method did not converge')
  }
  x
}

start.time.new <- proc.time()
newton(1, expression(cos(x)-x))
```

```
## [1] 0.7390851
```

```
total.time.new <- proc.time() - start.time.new
print(total.time.new)
```

```
##    user  system elapsed 
##       0       0       0
```

```
total.time.new - total.time.sec
```

```
##    user  system elapsed 
##       0       0       0
```

The secant is typically faster, e.g., by the above amount.

Problem 2:

```
craps <- function(ntrials){
  for(i in seq(ntrials)){
nloop <- 1
win <- 0
repeat{
  x <- sum(ceiling(6*runif(2)))
  if (nloop==1 && (x==7 | x == 11)) win <- 1
  if (nloop==1) roll <- x
  if (nloop > 1 && x == roll) win <- 1
  if (x==7 | x == 11 | win==1) break;
  nloop <- nloop +1
}
 if (i==1 && win==0) results <- 'LOSE :('
 if (i==1 && win==1) results <- 'WIN :D'
    if (i>1 && win==0) results <- c(results,'LOSE :(')
    if (i>1 && win==1) results <- c(results,'WIN :D')
  }
  return(results)
}


  set.seed(100)
  craps(3)
```

```
## [1] "LOSE :(" "LOSE :(" "LOSE :("
```

```
for(i in 1:1000){
set.seed(i)
y <- craps(10)
if (identical(y,rep("WIN :D",10))==TRUE) print(i)
}
```

```
## [1] 880
```

```
set.seed(880)
craps(10)
```

```
##  [1] "WIN :D" "WIN :D" "WIN :D" "WIN :D" "WIN :D" "WIN :D" "WIN :D"
##  [8] "WIN :D" "WIN :D" "WIN :D"
```

```
setwd("C:/Users/Jason/Documents")
# path: directory path to input files
# file: name of the output file; it should be written to path
# nTeams: number of teams in league
# cap: money available to each team
# posReq: number of starters for each position
# points: point allocation for each category
ffvalues <- function(path, file='outfile.csv', nTeams=12, cap=200, posReq=c(qb=1, rb=2, wr=3, te=1, k=1
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                              rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)) {
  ## read in CSV files
k <- read.csv(file.path('proj_k15.csv'), header=TRUE, stringsAsFactors=FALSE)
qb <- read.csv(file.path('proj_qb15.csv'), header=TRUE, stringsAsFactors=FALSE)
rb <- read.csv(file.path('proj_rb15.csv'), header=TRUE, stringsAsFactors=FALSE)
te <- read.csv(file.path('proj_te15.csv'), header=TRUE, stringsAsFactors=FALSE)
wr <- read.csv(file.path('proj_wr15.csv'), header=TRUE, stringsAsFactors=FALSE)
# generate unique list of column names
cols <- unique(c(names(k), names(qb), names(rb), names(te), names(wr)))
# create a new column in each data.frame
# values are recylcled
# concept: ?Extract
k[,'pos'] <- 'k'
qb[,'pos'] <- 'qb'
rb[,'pos'] <- 'rb'
te[,'pos'] <- 'te'
wr[,'pos'] <- 'wr'

# append 'pos' to unique column list
cols <- c(cols, 'pos')

# create common columns in each data.frame
# initialize values to zero
k[,setdiff(cols, names(k))] <- 0
qb[,setdiff(cols, names(qb))] <- 0
rb[,setdiff(cols, names(rb))] <- 0
te[,setdiff(cols, names(te))] <- 0
wr[,setdiff(cols, names(wr))] <- 0

# combine data.frames by row, using consistent column order
x <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])

# calculate new columns
# convert NFL stat to fantasy points
x[,'p_fg'] <- x[,'fg']*points["fg"]
x[,'p_xpt'] <- x[,'xpt']*points["xpt"]
x[,'p_pass_yds'] <- x[,'pass_yds']*points["pass_yds"]
x[,'p_pass_tds'] <- x[,'pass_tds']*points["pass_tds"]
```

```r
x[,'p_pass_ints'] <- x[,'pass_ints']*points["pass_ints"]
x[,'p_rush_yds'] <- x[,'rush_yds']*points["rush_yds"]
x[,'p_rush_tds'] <- x[,'rush_tds']*points["rush_tds"]
x[,'p_fumbles'] <- x[,'fumbles']*points["fumbles"]
x[,'p_rec_yds'] <- x[,'rec_yds']*points["rec_yds"]
x[,'p_rec_tds'] <- x[,'rec_tds']*points["rec_tds"]

# sum selected column values for every row
# this is total fantasy points for each player
x[,'points'] <- rowSums(x[,grep("^p_", names(x))])


  ## calculate dollar values

# create new data.frame ordered by points descendingly
x2 <- x[order(x[,'points'], decreasing=TRUE),]

# determine the row indeces for each position
k.ix <- which(x2[,'pos']=='k')
qb.ix <- which(x2[,'pos']=='qb')
rb.ix <- which(x2[,'pos']=='rb')
te.ix <- which(x2[,'pos']=='te')
wr.ix <- which(x2[,'pos']=='wr')

# calculate marginal points by subtracting "baseline" player's points
x2[k.ix, 'marg'] <- x2[k.ix,'points'] - x2[k.ix[12],'points']
x2[qb.ix, 'marg'] <- x2[qb.ix,'points'] - x2[qb.ix[12],'points']
x2[rb.ix, 'marg'] <- x2[rb.ix,'points'] - x2[rb.ix[24],'points']
x2[te.ix, 'marg'] <- x2[te.ix,'points'] - x2[te.ix[12],'points']
x2[wr.ix, 'marg'] <- x2[wr.ix,'points'] - x2[wr.ix[36],'points']

# create a new data.frame subset by non-negative marginal points
x3 <- x2[x2[,'marg'] >= 0,]

# re-order by marginal points
x3 <- x3[order(x3[,'marg'], decreasing=TRUE),]

# reset the row names
rownames(x3) <- NULL

# calculation for player value
x3[,'value'] <- x3[,'marg']*(12*200-nrow(x3))/sum(x3[,'marg']) + 1

# create a data.frame with more interesting columns
x4 <- x3[,c('PlayerName','pos','points','marg','value')]


  ## save dollar values as CSV file
write.csv(file=file,x4)

  ## return data.frame with dollar values
return(x4)
}
```

```r
x1 <- ffvalues('.')
x1[which(x1$value > 20),]
```

```
##           PlayerName pos  points   marg    value
## 1     Marshawn Lynch  rb 213.360 87.855 87.33094
## 2    Adrian Peterson  rb 210.010 84.505 84.03906
## 3          Eddie Lacy  rb 209.260 83.755 83.30207
## 4     Jamaal Charles  rb 208.470 82.965 82.52577
## 5        Andrew Luck  qb 326.470 68.048 67.86754
## 6   Demaryius Thomas  wr 137.720 65.040 64.91173
## 7      C.J. Anderson  rb 189.915 64.410 64.29265
## 8       Le'Veon Bell  rb 189.570 64.065 63.95364
## 9      Aaron Rodgers  qb 321.682 63.260 63.16260
## 10     Antonio Brown  wr 135.210 62.530 62.44527
## 11        Dez Bryant  wr 132.980 60.300 60.25395
## 12        Matt Forte  rb 185.745 60.240 60.19499
## 13      LeSean McCoy  rb 183.910 58.405 58.39183
## 14 Odell Beckham Jr.  wr 130.930 58.250 58.23951
## 15    DeMarco Murray  rb 182.375 56.870 56.88345
## 16       Jeremy Hill  rb 182.250 56.745 56.76062
## 17    Rob Gronkowski  te 112.210 55.210 55.25225
## 18    Calvin Johnson  wr 125.170 52.490 52.57943
## 19      Randall Cobb  wr 124.865 52.185 52.27973
## 20       Julio Jones  wr 118.555 45.875 46.07919
## 21    Russell Wilson  qb 300.048 41.626 41.90390
## 22    Peyton Manning  qb 297.986 39.564 39.87767
## 23    Alshon Jeffery  wr 112.075 39.395 39.71160
## 24        A.J. Green  wr 111.550 38.870 39.19571
## 25       Mark Ingram  rb 162.255 36.750 37.11248
## 26      Lamar Miller  rb 162.125 36.620 36.98474
## 27        Drew Brees  qb 294.826 36.404 36.77249
## 28    Justin Forsett  rb 161.735 36.230 36.60150
## 29        Mike Evans  wr 107.635 34.955 35.34862
## 30      Jimmy Graham  te  91.755 34.755 35.15209
## 31     Alfred Morris  rb 158.585 33.080 33.50615
## 32  Emmanuel Sanders  wr 103.485 30.805 31.27061
## 33       T.Y. Hilton  wr 103.435 30.755 31.22148
## 34     Melvin Gordon  rb 152.570 27.065 27.59549
## 35       Carlos Hyde  rb 152.015 26.510 27.05012
## 36        Frank Gore  rb 150.675 25.170 25.73337
## 37         Matt Ryan  qb 282.788 24.366 24.94331
## 38     Brandin Cooks  wr  95.550 22.870 23.47326
## 39   Latavius Murray  rb 148.110 22.605 23.21286
## 40   Jordan Matthews  wr  93.315 20.635 21.27704
```

```r
dim(x1[which(x1$value > 20),])
```

```
## [1] 40  5
```

40 players are worth more than 20 dollars.

```
setwd("C:/Users/Jason/Documents")
x1[which(x1$pos=="rb"),][15,]
```

```
##       PlayerName pos points   marg    value
## 34 Melvin Gordon  rb 152.57 27.065 27.59549
```

Melvin Gordon is the 15th most valuable running back.

```
setwd("C:/Users/Jason/Documents")
x2 <- ffvalues(getwd(), '16team.csv', nTeams=16, cap=150)
x2[which(x2$value > 20),]
```

```
##           PlayerName pos  points   marg    value
## 1     Marshawn Lynch  rb 213.360 87.855 87.33094
## 2    Adrian Peterson  rb 210.010 84.505 84.03906
## 3         Eddie Lacy  rb 209.260 83.755 83.30207
## 4     Jamaal Charles  rb 208.470 82.965 82.52577
## 5        Andrew Luck  qb 326.470 68.048 67.86754
## 6   Demaryius Thomas  wr 137.720 65.040 64.91173
## 7      C.J. Anderson  rb 189.915 64.410 64.29265
## 8       Le'Veon Bell  rb 189.570 64.065 63.95364
## 9      Aaron Rodgers  qb 321.682 63.260 63.16260
## 10     Antonio Brown  wr 135.210 62.530 62.44527
## 11        Dez Bryant  wr 132.980 60.300 60.25395
## 12        Matt Forte  rb 185.745 60.240 60.19499
## 13      LeSean McCoy  rb 183.910 58.405 58.39183
## 14 Odell Beckham Jr.  wr 130.930 58.250 58.23951
## 15    DeMarco Murray  rb 182.375 56.870 56.88345
## 16       Jeremy Hill  rb 182.250 56.745 56.76062
## 17    Rob Gronkowski  te 112.210 55.210 55.25225
## 18    Calvin Johnson  wr 125.170 52.490 52.57943
## 19      Randall Cobb  wr 124.865 52.185 52.27973
## 20       Julio Jones  wr 118.555 45.875 46.07919
## 21    Russell Wilson  qb 300.048 41.626 41.90390
## 22    Peyton Manning  qb 297.986 39.564 39.87767
## 23    Alshon Jeffery  wr 112.075 39.395 39.71160
## 24        A.J. Green  wr 111.550 38.870 39.19571
## 25       Mark Ingram  rb 162.255 36.750 37.11248
## 26      Lamar Miller  rb 162.125 36.620 36.98474
## 27        Drew Brees  qb 294.826 36.404 36.77249
## 28    Justin Forsett  rb 161.735 36.230 36.60150
## 29        Mike Evans  wr 107.635 34.955 35.34862
## 30      Jimmy Graham  te  91.755 34.755 35.15209
## 31     Alfred Morris  rb 158.585 33.080 33.50615
## 32  Emmanuel Sanders  wr 103.485 30.805 31.27061
## 33       T.Y. Hilton  wr 103.435 30.755 31.22148
## 34     Melvin Gordon  rb 152.570 27.065 27.59549
## 35       Carlos Hyde  rb 152.015 26.510 27.05012
## 36        Frank Gore  rb 150.675 25.170 25.73337
## 37         Matt Ryan  qb 282.788 24.366 24.94331
## 38     Brandin Cooks  wr  95.550 22.870 23.47326
## 39   Latavius Murray  rb 148.110 22.605 23.21286
## 40   Jordan Matthews  wr  93.315 20.635 21.27704
```

```
dim(x2[which(x2$value > 20),])
```

```
## [1] 40  5
```

40 players are worth more than 20 dollars.

```
setwd("C:/Users/Jason/Documents")
temp <- x2[1:40,]
temp[which(temp$pos=="wr"),]
```

```
##           PlayerName pos  points   marg    value
## 6   Demaryius Thomas  wr 137.720 65.040 64.91173
## 10     Antonio Brown  wr 135.210 62.530 62.44527
## 11        Dez Bryant  wr 132.980 60.300 60.25395
## 14 Odell Beckham Jr.  wr 130.930 58.250 58.23951
## 18    Calvin Johnson  wr 125.170 52.490 52.57943
## 19      Randall Cobb  wr 124.865 52.185 52.27973
## 20       Julio Jones  wr 118.555 45.875 46.07919
## 23    Alshon Jeffery  wr 112.075 39.395 39.71160
## 24        A.J. Green  wr 111.550 38.870 39.19571
## 29        Mike Evans  wr 107.635 34.955 35.34862
## 32  Emmanuel Sanders  wr 103.485 30.805 31.27061
## 33       T.Y. Hilton  wr 103.435 30.755 31.22148
## 38     Brandin Cooks  wr  95.550 22.870 23.47326
## 40   Jordan Matthews  wr  93.315 20.635 21.27704
```

```
dim(temp[which(temp$pos=="wr"),])
```

```
## [1] 14  5
```

14 wide receivers are in the top 40.

```
setwd("C:/Users/Jason/Documents")
x3 <- ffvalues('.', 'qbheavy.csv', posReq=c(qb=2, rb=2, wr=3, te=1, k=0),
        points=c(fg=0, xpt=0, pass_yds=1/25, pass_tds=6, pass_ints=-2,
                rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6))

x3[which(x3$value > 20),]
```

```
##           PlayerName pos  points   marg    value
## 1     Marshawn Lynch  rb 213.360 87.855 88.59253
## 2    Adrian Peterson  rb 210.010 84.505 85.25254
## 3         Eddie Lacy  rb 209.260 83.755 84.50478
## 4     Jamaal Charles  rb 208.470 82.965 83.71714
## 5        Andrew Luck  qb 397.070 82.470 83.22362
## 6      Aaron Rodgers  qb 390.682 76.082 76.85470
## 7   Demaryius Thomas  wr 137.720 65.040 65.84569
## 8      C.J. Anderson  rb 189.915 64.410 65.21757
## 9       Le'Veon Bell  rb 189.570 64.065 64.87360
## 10     Antonio Brown  wr 135.210 62.530 63.34319
## 11        Dez Bryant  wr 132.980 60.300 61.11985
## 12        Matt Forte  rb 185.745 60.240 61.06003
## 13      LeSean McCoy  rb 183.910 58.405 59.23051
## 14 Odell Beckham Jr.  wr 130.930 58.250 59.07598
## 15    DeMarco Murray  rb 182.375 56.870 57.70010
## 16       Jeremy Hill  rb 182.250 56.745 57.57547
## 17    Rob Gronkowski  te 112.210 55.210 56.04506
## 18    Peyton Manning  qb 368.386 53.786 54.62531
## 19    Calvin Johnson  wr 125.170 52.490 53.33318
## 20      Randall Cobb  wr 124.865 52.185 53.02910
## 21       Julio Jones  wr 118.555 45.875 46.73795
## 22        Drew Brees  qb 357.226 42.626 43.49865
## 23    Alshon Jeffery  wr 112.075 39.395 40.27731
## 24        A.J. Green  wr 111.550 38.870 39.75387
## 25       Mark Ingram  rb 162.255 36.750 37.64021
## 26      Lamar Miller  rb 162.125 36.620 37.51060
## 27    Justin Forsett  rb 161.735 36.230 37.12176
## 28    Russell Wilson  qb 349.848 35.248 36.14270
## 29        Mike Evans  wr 107.635 34.955 35.85057
## 30      Jimmy Graham  te  91.755 34.755 35.65117
## 31     Alfred Morris  rb 158.585 33.080 33.98117
## 32  Emmanuel Sanders  wr 103.485 30.805 31.71297
## 33       T.Y. Hilton  wr 103.435 30.755 31.66312
## 34     Melvin Gordon  rb 152.570 27.065 27.98414
## 35       Carlos Hyde  rb 152.015 26.510 27.43080
## 36         Matt Ryan  qb 340.388 25.788 26.71096
## 37        Frank Gore  rb 150.675 25.170 26.09480
## 38     Brandin Cooks  wr  95.550 22.870 23.80167
## 39   Latavius Murray  rb 148.110 22.605 23.53747
## 40   Jordan Matthews  wr  93.315 20.635 21.57335
```

```
dim(x3[which(x3$value > 20),])
```

```
## [1] 40  5
```

40 players are worth more than 20 dollars.

```
setwd("C:/Users/Jason/Documents")
temp <- x3[1:30,]
temp[which(temp$pos=="qb"),]
```

```
##        PlayerName pos  points   marg    value
## 5     Andrew Luck  qb 397.070 82.470 83.22362
## 6   Aaron Rodgers  qb 390.682 76.082 76.85470
## 18 Peyton Manning  qb 368.386 53.786 54.62531
## 22     Drew Brees  qb 357.226 42.626 43.49865
## 28 Russell Wilson  qb 349.848 35.248 36.14270
```

```
dim(temp[which(temp$pos=="qb"),])
```

```
## [1] 5 5
```

5 quarterbacks are in the top 40.

Problem 4:

```
objs <- mget(ls("package:base"), inherits = TRUE)
funs <- Filter(is.function, objs)

n.args <- length(as.list(args(names(funs)[1])))-1
for(i in 2:length(funs)){
 n.args <- c(n.args,length(as.list(args(names(funs)[i])))-1)
}
n.args <- matrix(n.args)
colnames(n.args) <- "numb.args"
n.args <- cbind(c(1:length(n.args)),n.args)
funs[n.args[which(n.args[,2]==max(n.args[,2])),][1]]
```

```
## $scan
## function (file = "", what = double(), nmax = -1L, n = -1L, sep = "",
##     quote = if (identical(sep, "\n")) "" else "'\"", dec = ".",
##     skip = 0L, nlines = 0L, na.strings = "NA", flush = FALSE,
##     fill = FALSE, strip.white = FALSE, quiet = FALSE, blank.lines.skip = TRUE,
##     multi.line = TRUE, comment.char = "", allowEscapes = FALSE,
##     fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
## {
##     na.strings <- as.character(na.strings)
##     if (!missing(n)) {
##         if (missing(nmax))
##             nmax <- n/pmax(length(what), 1L)
##         else stop("either specify 'nmax' or 'n', but not both.")
##     }
##     if (missing(file) && !missing(text)) {
##         file <- textConnection(text, encoding = "UTF-8")
##         encoding <- "UTF-8"
##         on.exit(close(file))
##     }
##     if (is.character(file))
##         if (file == "")
##             file <- stdin()
##         else {
##             file <- if (nzchar(fileEncoding))
##                 file(file, "r", encoding = fileEncoding)
##             else file(file, "r")
##             on.exit(close(file))
##         }
##     if (!inherits(file, "connection"))
##         stop("'file' must be a character string or connection")
##     .Internal(scan(file, what, nmax, sep, dec, quote, skip, nlines,
##         na.strings, flush, fill, strip.white, quiet, blank.lines.skip,
##         multi.line, comment.char, allowEscapes, encoding, skipNul))
## }
## <bytecode: 0x000000000a298928>
## <environment: namespace:base>
```

Scan has the largest number of arguments (22).

```r
length(n.args[which(n.args[,2] < 1),])
```

```
## [1] 146
```

146 do not have arguments.